ALGORITHM 82

ECONOMISING A SEQUENCE 2

BRIAN H. MAYOH

Digital Computer Laboratory, University of Illinois, Urbana, Ill.

```
procedure ECONOMISER 2 (desired property, costs, n, C, r,
  Reject list); Boolean procedure desired property;
  integer n, r; array costs; Boolean array Reject list;
begin comment In some applications of ECONOMISER 1, it
  is simple to establish that some subsequences are redundant in
  the sense that any sequence containing them is certainly not
  the cheapest subsequence with the desired property. For such
  applications ECONOMISER 2 avoids all unnecessary calls of
  desired property. The new formal parameters are: r a variable
  whose value is initially 0 and is increased by 1 every time that
  desired property discovers a new redundant subsequence.
  Reject list an array of size [1:r,1:n]. Reject list [a,b] carries the
  answer to: Is element b of the original sequence in the a^th
  redundant subsequence found by desired property?;
  real i; integer d, j, k, ℓ; Boolean gapfilled, first time;
  procedure INSIDE (entrymaker); Boolean entrymaker;
  begin own real array prices[1:d];
        own Boolean array alternatives[1:d,1:n];
        procedure ENTER SUCCESSORS;
        begin integer c; Boolean array ssq[1:n];
            for j := 1 step 1 until n do ssq[j] := C[j];
            c := n−1;
        A:  if ¬ ssq[c] then begin c := c−1; go to A end;
            C[c] := false; C[c+1] := true;
            INSIDE (true);
            gapfilled := true;
        B:  c := c−1;
            go to if c=0 then F else if ssq[c] then
              (if c=1 then F else B) else if c=1 then
              E else if ssq[c−1] then D else F;
        D:  ssq[c−1] := false;
        E:  for j := 1 step 1 until n do C[j] := ssq[j] ≡ j≠c;
            INSIDE (true);
        F:  end of ENTER SUCCESSORS;
        if entrymaker then
        begin for j := 1 step 1 until r do
            begin for k := 1 step 1 until n do
              begin if ¬ C[k] ∧ Reject list[j,k] then
                go to G end;
              ENTER SUCCESSORS; go to H;
        G:  end;
            i := 0; if gapfilled then d := d+1;
            for j := 1 step 1 until n do
            begin alternatives[if gapfilled then
              d else ℓ, j] := C[j];
              if C[j] then i := i + costs[j]
            end; prices[if gapfilled then d else ℓ] := i
        end; if first time ∨ ¬ entrymaker then
        begin i := 0; gapfilled := first time := false;
          for j := 1 step 1 until d do
          begin if prices[j] < i then
            begin ℓ := j; i := prices[ℓ] end
          end;
            for j := 1 step 1 until n do
              C[j] := alternatives[ℓ,j];
            if desired property then go to found;
            ENTER SUCCESSORS; go to reenter
            end;
        H:  end of INSIDE;
        for j := 1 step 1 until n do C[j] := j=1;
        d := 0; first time := gapfilled := true;
reenter: INSIDE (first time);
found:
end of ECONOMISER 2;
```